Express Mail Label No. EM443447383US

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No.	CTX-016CN (1545/27)
First Named Inventor	Pedersen
Title	System And Method For Transmitting Data From A Server Application To More Than One Client Node

APPLICATION ELEMENTS	ACCOMPANYING APPLICATION PARTS
ADDRESS TO: **Box Patent Application, Assistant Commissioner for Patents, Washington, D.C. 20231**	
1. ☒ Fee Transmittal Form	
2. ☒ Specification and Drawings [Total Pages 33] - Specification - (21 pages) - Claims - (3 pages) - Abstract - (2 pages) - Sheets of Drawings - (7 sheets) ☒ Formal ☐ Informal	7. ☐ 37 CFR 3.73(b) Statement (when there is an assignee) ☐ Power of Attorney
	8. ☐ English Translation Document *(if applicable)*
	9. ☐ Information Disclosure Statement (IDS)/PTO-1449 ☐ Copies of IDS Citations
3. ☒ Oath or Declaration [Total Pages 4] a. ☐ Newly executed (original) b. ☒ Copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional with Box 17 completed)* ***[Note Box 4 below]***	10. ☐ Preliminary Amendment ☐ Drawings [Total Sheets] ☐ Letter to Official Draftsperson Including Drawings [Total Pages]
	11. ☒ Return Receipt Postcard
4. ☐ Incorporation by Reference (usable if Box 3b is checked) The entire Disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 3b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.	12. ☐ Small Entity Statement(s) ☐ Statements filed in prior application, (Status still proper and desired)
	13. ☐ Certified Copy of Priority Document(s)
5. ☐ Microfiche Computer Program *(Appendix)*	14. ☐ Deletion of Inventor(s) Signed statement attached deleting inventor(s) named in the prior application.
6. ☐ Nucleotide and/or Amino Acid Sequence Submission ☐ Computer Readable Copy ☐ Paper Copy (identical to computer copy) ☐ Statement verifying identify of above copies	15. ☐ Patent Application Data Entry Form
	16. ☒ Other: Associate Power of Attorney

17. ☒ **If a CONTINUING APPLICATION,** check appropriate box and supply the requisite information:
☒ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application Serial No. 08/856,051.
Priority to the above application(s) is claimed under 35 U.S.C. 120.
Prior application information: Examiner: C. Ho. Group/Art Unit: 2757.

18. ☐ **Priority - 35 U.S.C. 119**
☐ Priority of application Serial No. _____ filed on _____ in _____ is claimed under 35 U.S.C. 119.
☐ The certified copy has been filed in prior U.S. application Serial No. _____/_____ on _____.
☐ The certified copy will follow.

CORRESPONDENCE ADDRESS	SIGNATURE BLOCK
Direct all correspondence to: Patent Administrator Testa, Hurwitz & Thibeault, LLP High Street Tower 125 High Street Boston, MA 02110 Tel. No.: (617) 248-7000 Fax No.: (617) 248-7100	Respectfully submitted, Date: May 13, 1999 Reg. No. 41,274 Tel. No.: (617) 248-7501 Fax No.: (617) 248-7100 Michael A. Rodriguez Agent for Applicants Testa, Hurwitz & Thibeault, LLP High Street Tower 125 High Street Boston, MA 02110

458tpl7557/33.A791446-1

EXPRESS MAIL LABEL NO. EM443447383US

APPLICATION

FOR

UNITED STATES

LETTERS PATENT

SPECIFICATION

(For Attorney Docket No. CTX-016)

TO ALL WHOM IT MAY CONCERN:

Be it known that I, **Bradley J. Pedersen**, a citizen of the United States of America, residing at 7700 S. Woodridge Drive, Parkland, Florida 33067, in the United States of America, has invented new and useful improvements in

SYSTEM AND METHOD FOR TRANSMITTING DATA FROM A SERVER APPLICATION TO MORE THAN ONE CLIENT NODE

of which the following is a specification.

SYSTEM AND METHOD FOR TRANSMITTING DATA FROM A SERVER APPLICATION TO MORE THAN ONE CLIENT NODE

Field of the Invention

The present invention relates generally to a system and method for communicating between a server application and multiple client nodes and more specifically to a system and method for transmitting the same data to more than one client node substantially simultaneously.

Background of the Invention

Shadowing (transmitting data destined for one client node substantially simultaneously to a second client node) and broadcasting (transmitting the same data substantially simultaneously to more than one client node) typically has been performed using a specialized transmitting application on a server node and specialized receiver applications on each of the client nodes. Shadowing is useful in monitoring data traffic and for creating a redundant copy of information being transmitted for data integrity and system security purposes. Broadcasting is useful in providing the same information to many users, when such information is "real-time" or when the information does not have a per se beginning or ending. For example, a stock price quotation program simply transmits the current prices of various stocks on a given exchange and the list repeats with the latest prices once the list of stocks is exhausted. Thus it is irrelevant to a user that he or she does not specify to the quotation program where to begin the list.

Such programs typically are written with a broadcast program in mind and require specialized receiver programs to receive the data transmitted. If an application has not been written as a broadcast program, the data transmitted by such an application can not typically be broadcast to multiple client nodes.

The present invention attempts to overcome this problem by permitting programs not written for broadcast functionality to be used to broadcast data over a network.

Summary of the Invention

The invention relates to a system and method for transmitting the same data to more than one client node substantially simultaneously. In one embodiment the invention relates to a method for transmitting the same data substantially simultaneously from an application executing on a server node to at least two client nodes executing a generalized receiver program. The method includes the steps of establishing a connection between a first client node and a first client protocol stack on the server node; establishing a connection between the application executing on the server node and the first client protocol stack; associating a first minimal communications protocol stack with the first client protocol stack; establishing a connection between the application executing on the server node and the first minimal communications protocol stack; establishing a connection between a second client node and a second client protocol stack on the server node; associating a second minimal communications protocol stack with the second client protocol stack; providing a connection between the first minimal protocol stack and the second minimal protocol stack; providing a connection between the second minimal protocol stack and said the second client protocol stack; and transmitting data from the application program to the first client protocol stack and the first minimal protocol stack, substantially simultaneously.

The invention also relates to a communication system including a server and two or more client nodes. In one embodiment the server node comprises an application program; a first client protocol stack in electrical communication with the application program; a first minimal protocol stack in electrical communication with the application program; a second minimal protocol stack in electrical communication with the first minimal protocol stack; and a second client protocol stack in electrical communication with the second minimal protocol stack. In addition the system includes a first client node in electrical communication with the first client protocol stack and a second client node in electrical communication with the second client protocol stack. Data from the application program is transmitted to the client protocol stack and the first minimal protocol stack substantially simultaneously.

Brief Description of the Drawings

The foregoing and other objects, features and advantages of the invention will become apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings.

FIG. 1 is a highly schematic diagram of an embodiment of a communication system utilizing the invention;

FIG. 2 is a block diagram of an embodiment of the invention showing the connections between various components of the server of Fig. 1 which occur during communication between the clients and server;

FIG. 3 is a block diagram of an embodiment of the invention that maintains and manages multiple client node connections;

FIG. 4 is a block diagram of an embodiment of the system for embedding applications in an HTML page;

FIG. 5 is a diagrammatic view of a client node;

FIG. 6 is a block diagram of an embodiment of the invention depicting the use of a multiplexer to transmit the same data from an application to more than one client; and

FIG. 7 is a block diagram of the embodiment of the invention in which the broadcast capabilities are increased by fan out.

Detailed Description Of The Invention

Referring now to FIG. 1, in brief overview, a typical network 20 includes at least one client node 24, at least one server node 34, 34', and a master network information node 40 connected together by a communications link 44. The embodiment shown in Fig. 1 depicts the communications link 44 as a local area network ring or LAN ring, but any communication topology may be used. For the purpose of explanation the server node 34 is assumed to have the application 30 requested by the client node 24. Also, for the purpose of explanation, the master network information node 40 is assumed to be a distinct server node, but in actuality the master network information node 40 may be an application execution server node 34. It should be noted that on a given LAN several nodes may be capable of acting as a network information node, but at any one time only one of such nodes is designated the master network information node 40 for the system 20 and it is to this node that client requests for server information are directed.

The master network information node 40 maintains a table of addresses for the application execution server nodes 34, 34'. In addition, the master network information node 40 receives messages from each application execution server node 34, 34' indicating its level of activity. The

level of activity of the application execution server nodes 34, 34' is maintained in a table along with the address of each of the application execution server nodes 34 and is used by the communications system 44 for load leveling.

When the client 24 wishes to have an application executed on an application execution server node 34, the client node 24 sends a request to the general communications port previously defined by the communications protocol or to the "well-known" communications port on the master network information node 40. In one embodiment the communication takes place by way of a datagram service. The master network information node 40 accesses the table of server addresses and returns a message containing the address of the application execution server or application server 34 which has the requested application and also which has the least load. Subsequent communications are automatically addressed by the client also to a "well-known" or predefined general communications port on the server node 34. In one embodiment, the type of protocol with which the initial query was made to the master network information node 40 determines the protocol of the information returned by the master network information node 40 to the client node 24. Thus if the request were made using a TCP/IP datagram, the master network information node 40 would return the TCP/IP address of the server 34 to the client node 24 and the client node 24 would subsequently establish contact with the server node 34 using that protocol. In another embodiment, the datagram requesting an application address by a client 24 includes a request for a different type of protocol than the one used to send the request to the master network information node 40. For example, the client 24 may make a request to the master network information node 40 using the IPX protocol and request the address of the application server as a TCP/IP protocol address.

When a client node 24 (actually a client process 56 on a client node 24) desires to communicate with an application on a server node 34, 34' the client node 24 begins by issuing a network request to determine the location of the server 34 having the desired application. This request is received by the master network information node 40 (also referred to as a network browser 40) residing somewhere on the network. In this Fig. 1, the network browser 40 is shown for simplicity as residing on a different server 40 from the server which has the application, but such may generally not be the case.

The network master information node 40 returns the network address of the server node 34 having the desired application 30 to the client node 24. The client node 24 then uses the information received from the network master information node 40 to request connection to the application executing on the specified server 34. As is described above, such a connection is first established to a "well-known" communications port and is later transferred to a specific communications port under control of a connection manager. The specific communications port is associated with the application executing on the server node 34 which then communicates with the client node 24 through the specific communications port.

In more detail, and referring to Fig. 2, the client process 56 on client node 24 makes a request 54 to the network master information node 40 to obtain the address of a server node 34 which includes the desired application 62. The network master information node 40 returns to the client node 24 a message 58 containing the address of the server node 34 which includes the server application 62. In one embodiment, the protocol used at this point of the connection is a datagram service.

The client node 24 uses the returned address to establish a communication channel 68 with the server 34. The port number used by the client 24 corresponds to the "well-known port" in the server 34 which has been defined by the network protocol as the port by which the server 34 establishes communication connections with clients 24. The well-known port 72 has a rudimentary protocol stack 76 which includes primarily an endpoint data structure 78.

The endpoint data structure 78 points to the communication protocol stack 76 and client connection thereby establishing a unique representation or "handle" for the client 24. The endpoint data structure 78 permits the connection between the server 34 and the client 24 to be moved at will between the connection manager 80 and the various applications 62 on the server 34. The endpoint data structure 78, in one embodiment, not only contains the handle to the client 24 but may also contain other information relating to the client connection. In the embodiment shown, the application server 34 monitors activity on a specific communications system (e.g. LAN or WAN) and has initialized this minimum protocol stack 76 with only the necessary protocol modules needed to support a "TTY" communication mode. The "TTY" communication mode is a simple ASCII stream with no protocol assumptions above the transport layer. That is, there are no protocol layers for compression, encryption, reliability, framing, or presentation of transmitted data. Thus a client node 24 seeking an application 62 running on the server 34 establishes a connection to the well-known communications port 72 with the minimum protocol set needed to support a TTY communication mode.

A communications manager 80 executing on the server node 34 is "listening" to the well-known communications port 72 for a connection request 68. When a connection request 68 is

received from the client node 24, the connection manager 80 is notified 84. The connection manager 80 knows which protocol is being used based on the notification 84.

With this information the connection manager 80 creates a new minimum protocol communications stack 104, starts the execution environment 96 and binds the new minimum protocol stack 104 to the execution environment 96. In one embodiment, the server 34 includes a number of execution environments 96 which have been previously been started, but which have not been associated with a communications port. In this embodiment, the pre-connection starting of the execution environments permits a faster response time than if each execution environment 96 is started when the connection request is received from the client 24. When the execution environment 96 is started, the server application 62 requested by the client 24 is also started. In another embodiment, if the client 24 does not specify an application, either a default application is started or simply the execution environment 96 with no application is started.

The communications manager 80 then moves the client connection, including the unique client identifier or handle, from the well-known port 76 to the new minimum protocol stack 104. The communications manager 80, using the minimum protocol stack sends a TTY data stream that indicates service is available. Thus, this method for detecting a client connection is independent of the port to which the connection is first established. If the client node 24 does not respond within a prescribed time period (e.g. 5 seconds) to the service available message, a resends of the "service available" message is performed by the server 34.

If the client 24 receives the message, the client 24 sends a TTY string indicating that the "service available" message was detected. The client 24 waits for the server 34 to respond and if the response is not within a prescribed time interval (e.g. 5 seconds) the client 24 resends the

message. The connection manager 80 then queries 90 the client 24 asking for the client's default communication parameters. This query 90 takes the form of a message which is passed back to the client 24 and which indicates that the client 24 should respond with details regarding what protocols the client 24 would like to use in the connection.

In response, the client 24 sends a set of protocol packets 92; each packet of which is used to specify a required or optional protocol module that is being requested from the server 34. In one embodiment, the number of packets in the set is variable with one packet being sent for each protocol requested. In another embodiment, the number of packets that is being sent is included in the header of the first packet. In a third embodiment, the remaining number of packets being sent is included in the header of each packet and is decremented with each succeeding packet sent. Thus, the client 24 may respond to the query 90 by indicating that, for example, encryption and data compression will be used. In such a case, two protocol packets will be sent from the client 24 to the server 34 and, in one embodiment, the header of the first packet will indicate the number of packets as two.

Once the responses to the query 90 have been received, the connection manager 80 builds a protocol stack using protocol drivers 120, 120', 120'' which correspond to the protocols requested by the client node 24. In one embodiment, the communications manager 80 places each of the required protocol drivers 120, 120', 120'', corresponding to the requested client protocols (e.g. an encryption driver if encryption is desired by the client) into the protocol stack "container" 112 and links them together. This dynamic process allows a client node 24 to specify the contents of a protocol stack dynamically without requiring that the server 34 have a prior protocol stack description for a particular client node 24. Using this method, multiple clients 24 may be served

by a single server, even if the separate clients 24 have vastly differing requirements for the associated communications channel. In the embodiment shown, each client 24, 24', 24'' is associated with a respective communications protocol stack 104, 104' and 104''. Such dynamically extensible protocol stacks are described in more detail below and in United States Patent Application Serial No. 08/540,891, filed on October 11, 1995 and incorporated herein by reference.

In the embodiment just discussed, the "container" 112 is a user level or kernel level device driver, such as an NT device driver. This container driver provides ancillary support for the inner protocol modules or "drivers" (generally 120) which correspond to the protocol requirements of the client node 24. This ancillary support is in the form of helper routines that, for example, aid one protocol driver to transfer data to the next driver. Alternatively, in another embodiment each protocol driver is a complete user-level or kernel-level driver in itself.

Referring now to the embodiment depicted in FIG. 3, the communications manager 60 includes two main software modules: ICASRV.EXE 90 and ICAAPI.DLL 94. In the embodiment shown, ICASRV.EXE 90 is the server side of a client/server interface. ICASRV.EXE 90 manages all communications states and is, in one embodiment, implemented as a WINDOWS NT™ service. A second part of the connection manager 60 is ICAAPI.DLL 94. ICAAPI.DLL 94 establishes the connection with the client, establishes the protocols to be used and notifies ICASRV.EXE 90 of the completion of the protocol stack. In one embodiment, a third module CDMODEM.DLL 96 is linked to ICAAPI.DLL 94'. CDMODEM.DLL 96 is a module which ICAAPI.DLL 94' uses to communicate with modem devices.

The connection methodology described above can be used for a client 24 running a Web browser program. For the purposes of this specification, the user running the Web browser program will be referred to as the "viewing user." The terms "server" or "server node" will be used to refer to machines hosting HTML files or applications that may be executed. For example, a viewing user runs a Web browser on a client node and makes file requests via the HTTP protocol to servers. The servers respond by transmitting file data to the client via the HTTP protocol. The Web browser run on the client receives the transmitted data and displays the data as an HTML page to the viewing user.

In brief overview and referring to FIG. 4, an HTML file 64 located on a server 34' and constructed in accordance with an embodiment of the invention includes a generic embedded window tag 66. The generic embedded window tag 66 is any data construct which indicates to a browser 60 displaying the HTML file 64 that a generic embedded window 66' should be displayed at a particular location in the HTML page 64' described by the HTML file 64. The generic embedded window tag 66 may include additional information, such as height of the window, width of the window, border style of the window, background color or pattern in the window, which applications may be displayed in the window, how often the output display should be updated, or any other additional information that is useful to enhance display of the application output.

Some examples of generic embedded window tags that can be embedded in an HTML file follow.

ActiveX tag

```
<object classid="clsid:238f6f83-b8b4-11cf-8771-00a024541ee3"
        data="/ica/direct.ica" CODEBASE="/cab/wfica.cab"
        width=436 height=295>
        <param name="Start" value="Auto">
        <param name="Border" value="On">
</object>
```

Netscape Plugin tag

```
<embed src="http://www.citrix.com/ica/direct.ica"
        pluginspage="http://www.citrix.com/plugin.html"
        height=295 width=436 Start=Auto Border=On>
<embed>
```

JAVA tag

```
<applet code=JICA.class width=436 height=295>

        <param name=Address             value="128.4.1.64">
        <param name=InitialProgram      value=Microsoft Word 7.0>
        <param name=Start               value=Auto>
        <param name=Border              value=On>

</applet>
```

In each case above, the tag indicates that a window having a height of 295 pixels and a width of 436 pixels should be drawn to receive application output. Each tag also specifies that the application should automatically start execution and that the window in which the application output is displayed should be drawn with a border. The ActiveX and Netscape Plugin tags have the remote application parameters specified in the file "direct.ica" located in the directory "/ica." The JAVA tag specifies the remote application parameters directly. In the example above, the address of the server hosting the application is specified as well as the name of the application to be executed.

The browser application 60 accesses the HTML file 64 by issuing a request to a specific Uniform Resource Locator (URL) address. The server 34' hosting the HTML file 64 transmits the HTML file 64 data to the browser application 60, which displays text and translates any tags that are included in the HTML file 64. The browser application 60 displays the HTML file 64 data as an HTML page 64'. If a generic embedded window tag 66 is present in the HTML file 64, such as one of the tags described above, the browser 60 draws a blank window 66' in the displayed HTML page 64'.

Execution of the desired application 62' may commence immediately upon display of the HTML page 64' or execution may await some signal, e.g. a specified user input which indicates execution of the application 62' should begin. Once execution of the application 62' is commenced, the browser application 60 instantiates a parameter handler 40 associated with the application window 66'. The parameter handler 40 instance may be spawned as a child process of the browser application 60, as a peer process of the browser application 60, or as a Dynamically Linked Library ("DLL") associated with the browser application 60.

The browser application 60 passes any specific parameters associated with the application window 66' that were provided by the generic embedded window 66 tag to the parameter handler 40 instance. Additionally, the browser application 60 may pass the handle for the application window 66' to the parameter handler 40 instance or the parameter handler 40 instance may query the browser application 60 to retrieve the handle for the application window 66'. The parameter handler 40 instance also spawns a network executive 50. The network executive 50 may be spawned as a child process of the parameter handler 40 instance or as a peer process of the parameter handler 40 instance.

The parameter handler 40 instance forwards any specified application window 66' parameters to the network executive 50. Parameters which are not specified by the parameter handler 40 instance or the embedded generic window tag 66 may be set to default values. The network executive 50 may have certain parameter defaults hard-coded, or the network executive 50 may access a file which contains parameter defaults.

The network executive 50 creates its own application output window 66". The network executive 50 creates its application output window 66" as a child of the displayed application window 66' and displays its application output window 66" directly over the parent window 66' drawn by the browser application 60. Since the application output window 66" drawn by the network executive 50 is a child of the application window 66' drawn by the browser application 60, the application output window 66" inherits various properties of its parent including position information. Accordingly, the application output window 66" will follow the application window 66' as the viewing user scrolls the screen of the browser application 60 or performs other actions which vary the position of the application window 66'.

The network executive 50 also establishes a communications channel with the server 60 and invokes execution of the desired application 62' by the server 34" using the connection methodology described above. The network executive 50, which acts as the client in the above description, passes any parameters it received from the parameter handler 40 instantiation to the server, along with any necessary default values. If a parameter is not passed to the server, the server may request the parameter if it is a necessary parameter which has no default value, e.g. "user id," or it may provide a default value for the parameter, e.g. execution priority. The server 34" begins execution of the desired application program 62' and directs the output to the network

executive 50. The network executive 50 receives data from the application program 62' and displays the output data in its application output window 66". Since the application output window 66" is drawn on top of the application window 66' drawn by the browser application 60, the application output data is displayed in the HTML page 64'. As noted above, the application output window 66" drawn by the network executive 50 is a child of the application window 66' drawn by the browser application 60. This allows the application output window 66" to scroll as the HTML page 64' is scrolled.

The application output window 66" also receives input from the viewing user. Raw input data, e.g. a mouse click, is received into the application output window 66" by the network executive 50. The network executive 50 forwards the raw input data to the application 62' executing on the server 34". In this manner, the viewing user is able to interact with the application 62' via the HTML page 64'.

Referring now to FIG. 5, the viewing user uses a so-called "browser" program to display an HTML page 64' having an application window 66' on the screen 18 of the user's computer 14. The viewing user may invoke execution of an application program 62'. Typically this is done by the user utilizing a "point-and-click" interface, i.e. the viewing user uses a mouse 16 to manipulate a cursor 12 that is also displayed on the screen 18 of the viewing user's computer 14. Once the cursor 12 is over a particular portion of the HTML page 64', the viewing user signals by "clicking" a button 15 on the mouse 16. Alternatively, the viewing user may also signal by pressing a key on an associated keyboard 17, such as the "return" key. In other embodiments, the viewing user may not use a mouse 16 at all, but may instead use a touchpad, a trackball, a pressure-sensitive tablet and pen, or some other input mechanism for manipulating the cursor 12.

In another embodiment, the application window 66', or another portion of the HTML page 64', may define a "hot zone." When the viewing user moves the cursor 12 into the "hot zone," execution of the application 62' on the server 34" is started.

Once the viewing user has indicated that execution of the application 62' should commence, the browser application 60 instantiates a parameter handler 40 and passes the instantiation parameters associated with the applications window 66' by the generic embedded window tag 66. The parameter handler 40 instance spawns a network executive 50 and passes to it the parameters of the application window 66'. The network executive 50 determines which application 62' is to be invoked, and on what server 34" that application 62' resides. Generally this information is passed to it by the parameter handler 40 instance which gets it from the browser application 60 in the form of the generic embedded window tag 66, but the network executive 50 may need to query a master network information node 40 or other various servers, in order to determine which servers, if any, host the desired application 62'. The network executive 50 then begins execution of the application and displays the output of the application program 62' in the applications window 66' as described in detail above.

The network executive 50 continues to directly display application output in the applications output window 66" until the viewing user indicates that execution of the application 62' should stop, e.g. by closing the application window 66', or until the viewing user clicks on a tag indicating that a different HTML page should be displayed. When this occurs, execution of the application 62' can be terminated. It is preferred, however, is to "cache" the connection. In effect, the first parameter handler 40 instance is not immediately terminated. However, the

application 62' continues executing with a reduced priority level, i.e. in "background" mode, because the first parameter handles 40 no longer has "focus".

In general, it is desirable to accomplish connection caching by providing the parameter handler 40 source code with a globally accessible data structure for registering instances. For example, the parameter handler 40 may be provided with a globally accessible linked list data structure, data array, data table, or other data structure. Because the data structure is globally available, each instance of the parameter handler 40 is able to read and write the data structure. This allows each instance of the parameter handler 40 to "register" with every other instance by writing to the data structure to signal its existence.

For embodiments in which no other connection information is stored, a predetermined limit on the number of connections that may be cached at any one time can be set. In these embodiments if registration of an instance would result in an excess number of cached connections, one of the "cached" connections is removed, i.e. the parameter handler 40 instantiation associated with that connection is notified that it should terminate. Before termination, the parameter handler 40 notifies its associated network executive 50 that it should terminate. In turn, the network executive 50 closes its session with the server hosting the application program 62' and then terminates.

In embodiments in which other information is stored, the additional information may be used to more effectively manage the cached connections. For example, if a user has not actively viewed an HTML page 64' in a predetermined number of minutes, e.g. ten minutes, the parameter handler 40 instantiation is instructed to terminate, the session with the hosting server is terminated, and the parameter handler 40 instance removes its entry in the registry.

Cached connection information may be managed using any known cache management scheme. Connection entries may be discarded on a "first in, first out" basis, i.e. the oldest entry is discarded each time a new entry must be added. Alternatively, cached connection information entries may be discarded on a "least recently used" basis, which discards information relating to connections which have been used the least amount by the user. Other cache management techniques, such as random replacement, may also be used.

If the viewing user returns to a previous HTML page 64' having a cached connection, the network executive 50 associated with the HTML page 64' is returned to the foreground, i.e., it regains "focus", and processing of the associated application resumes at a normal priority level. If necessary, the network executive 50 re-establishes the connection with the application 62'. Although no output data is stored by the network executive 50 for cached connections, as soon as a connection is re-established for an applications window 66' the connection to the application 62' is re-established and the application 10 again writes directly to the applications window 66'.

Referring to Fig. 6, it should be noted that any client 24, 24', 24'', or in fact, all the clients (generally 24) attached to server 34 with the application 63 may be another server 34', 34''. In this manner, data transmitted by the application 63 is sent to other servers prior to being sent to client nodes 24. In this manner, data transmitted by the application 63 is transmitted to an ever increasing number of client nodes as this network fans out.

When each client 24 terminates its connection with the server 34, each client protocol stack (generally 104) and its associated minimal stack (generally 107) is destroyed. Similarly, the minimal protocol stack (generally 106) associated with the first client protocol stack 104 is also destroyed. When the last of the minimal 107 and second (and subsequent) client protocol stacks

104 has terminated, the configuration is as it was initially with only a first client communications protocol stack 104 associated with the execution environment 96. Note that until all the second and subsequent client protocol stacks 104 are terminated, the first client protocol stack 104 may not be destroyed, even if the first client 24 is no longer present.

As shown in Fig. 2, each execution environment 96 communicates with each protocol stack 104 through a multiplexer 121, 121', 121". Now referring also to Fig. 6, with the present invention it is possible for more than one client to receive data being transmitted to the first client 24, for example, in order to shadow or monitor the transmission of data from a server 34 or to broadcast data from a specialized broadcast application, such as a stock quotation application, from which the same data is broadcast or transmitted substantially simultaneously to a number of clients (generally 24).

In such a case, the first client 24 causes the specialized application 63 to execute and transmit its data to the client 24 as discussed previously. When a second client 24' requests access to the broadcast application 63, the connection manager 80 begins to construct the protocol stack 104' for the second client 24' as previously discussed with regard to the first client 24. However, because the application 63 is a broadcast application, the connection manager 80 recognizes that it need not start an additional execution environment 96 and instead takes the steps necessary to send the data from the broadcast application 63 to the second client 24' and any additional clients 24''.

First, the connection manager 80 creates a first minimal communications protocol stack 106 which it associates with a communications protocol stack 104 of the first client 24. The connection manager 80 next creates a second minimal protocol stack 107 and associates it with

the communications protocol stack 104' of the second client 24'. As each additional client 24'' requests access to the broadcast application 63, another minimal protocol stack 106' is created and associated with the first client protocol stack 104 and another minimal protocol stack 107' and client protocol stack 104'' is created for each new client 24''. The first client protocol stack 104 and all the minimal protocol stacks 106, 106' associated with the first client protocol stack 104, and each pair of client protocol stacks 104', 104'' and minimal protocol stacks 107, 107' associated with each additional client 24', 24'' are in communication by way of a multiplexer 121.

When multiplexer 121 is directing data to or receiving data from only one client 24, the multiplexer 121 is acting as a simple pass-through device. However, when there is more than one client 24, 24', 24'' receiving data from or transmitting data to a single application 63, each multiplexer (generally 121) takes on two additional configurations. In one configuration, the multiplexer 121' is configured to send application data to or receive data from both the first client protocol stack 104 and each of the minimal communications protocol stacks 106, 106' associated with it. In the second configuration the multiplexer 121" is configured to send data received by the minimal protocol stack 107, 107' to the client protocol stack 104', 104'', respectively, associated with it. In this embodiment, the mux 121 may receive input data directly from each client protocol stack 104, 104', 104".

The connection manager 80 connects the minimal protocol stacks 106, 106' associated with the first client 24 with the minimal protocol stacks 107, 107' respectively, of the second 24' and subsequent clients 24'' and instructs the multiplexer 121 to direct output from the application 63 to the communications protocol stack 104 of the first client 24 and its associated minimal protocol stacks 106, 106'. The multiplexer 121 is also instructed by the connection manager 80

to connect each second and subsequent client minimal protocol stack 107, 107' to its associated client protocol stack 104, 104', respectively. Data transmitted to the first client 24 by way of the first client protocol stack 104 is therefore also transmitted to the minimal protocol stacks 106, 106' associated with the first client 24 and hence to the second 24' and subsequent clients 24'' by way of their associated protocol stacks 104', 104'', respectively, and associated minimal protocol stacks 107, 107', respectively. In one embodiment, the protocol stack container includes a data structure to keep track of the number and type of protocols associated with a given application 63.

Referring to Fig. 7, as discussed above, it is possible that the "clients" of one server 34 be other servers 34' and 34'' (only two being shown for simplicity). The second servers 34' and 34'' then transmit the data to clients (generally 24) or to additional servers. In this embodiment the output of the server protocol stack (generally 104) is connected to the protocol stacks 107' of the secondary servers 34', 34''. Then as described previously, the data is transmitted between the protocol stacks and out to the clients (generally 24). In this manner the data may fan out and be distributed to many more clients than may reasonably be supported by one server.

While the invention has been particularly shown and described with reference to specific preferred embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein departing from the spirit and scope of the invention as defined by the appended claims.

Claims

What is claimed is:

1. A method for transmitting the same data substantially simultaneously from an application executing on a server node to at least two client nodes, each client node executing a generalized receiver program, the method comprising the steps of:

(a) providing a connection between a first client node and a first client protocol stack on said server node;

(b) providing a connection between said application executing on said server node and said first client protocol stack;

(c) providing a connection between said application executing on said server node and a first minimal communications protocol stack;

(d) providing a connection between a second client node and a second client protocol stack on said server node;

(e) providing a connection between said first minimal protocol stack and a second minimal protocol stack;

(f) providing a connection between said second minimal protocol stack and said second client protocol stack; and

(g) transmitting data from said application program to said first client protocol stack and said first minimal protocol stack substantially simultaneously.

2. The method of claim 1 wherein said connection between said first client protocol stack and said application program occurs through a multiplexer.

3. The method of claim 1 wherein said connection between said first minimal protocol stack and said application program occurs through a multiplexer.

4. The method of claim 1 wherein said connection between said second client protocol stack and said second minimal protocol stack occurs through a multiplexer.

5. The method of claim 1 further comprises the step of associating a first minimal communications protocol stack with said first client protocol stack.

6. The method of claim 1 further comprising the step of associating a second minimal communications protocol stack with said second client protocol stack.

7. The method of claim 1 further comprising the step of determining whether said application program is suitable for broadcast.

8. A communication system comprising:

a server node comprising:

an application program;

a first client protocol stack in electrical communication with said application program;

a first minimal protocol stack in electrical communication with said application program;

a second minimal protocol stack in electrical communication with said first minimal protocol stack; and

a second client protocol stack in electrical communication with said second minimal protocol stack;

a first client in electrical communication with said first client protocol stack; and

a second client in electrical communication with said second client protocol stack;

whereby said data from said application program is transmitted to said first client protocol stack and said first minimal protocol stack substantially simultaneously.

SYSTEM AND METHOD FOR TRANSMITTING DATA FROM A SERVER APPLICATION TO MORE THAN ONE CLIENT NODE

Abstract of the Invention

The invention relates to a system and method for transmitting the same data to more than one client node substantially simultaneously. In one embodiment the invention relates to a method for transmitting the same data substantially simultaneously from an application executing on a server node to at least two client nodes. The method includes the steps of providing a connection between a first client node and a first client protocol stack and between the application and the first client protocol stack; associating a first minimal communications protocol stack with the first client protocol stack; providing a connection between the application and the first minimal communications protocol stack and between a second client node and a second client protocol stack; associating a second minimal communications protocol stack with the second client protocol stack; providing a connection between the first minimal protocol stack and the second minimal protocol stack; and between the second minimal protocol stack and said the client protocol stack. The method then transmits data from the application program to the first client protocol stack and the first minimal protocol stack substantially simultaneously.

The invention also relates to a communication system including a server node including: an application program, a first client protocol stack in electrical communication with the application program, a first minimal protocol stack in electrical communication with the application program; a second minimal protocol stack in electrical communication with the first minimal protocol stack; and a second client protocol stack in electrical communication with the second minimal protocol stack. In addition the system includes a first client in electrical communication with the first client protocol stack and a second client in electrical communication

with the second client protocol stack. The data from said application program is transmitted to the client protocol stack and the first minimal protocol stack substantially, simultaneously.

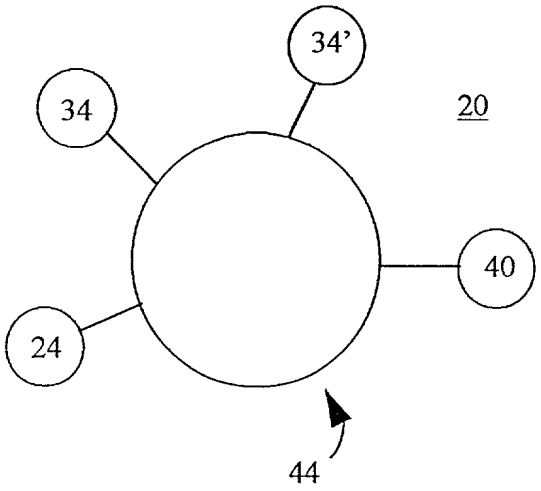

Fig. 1

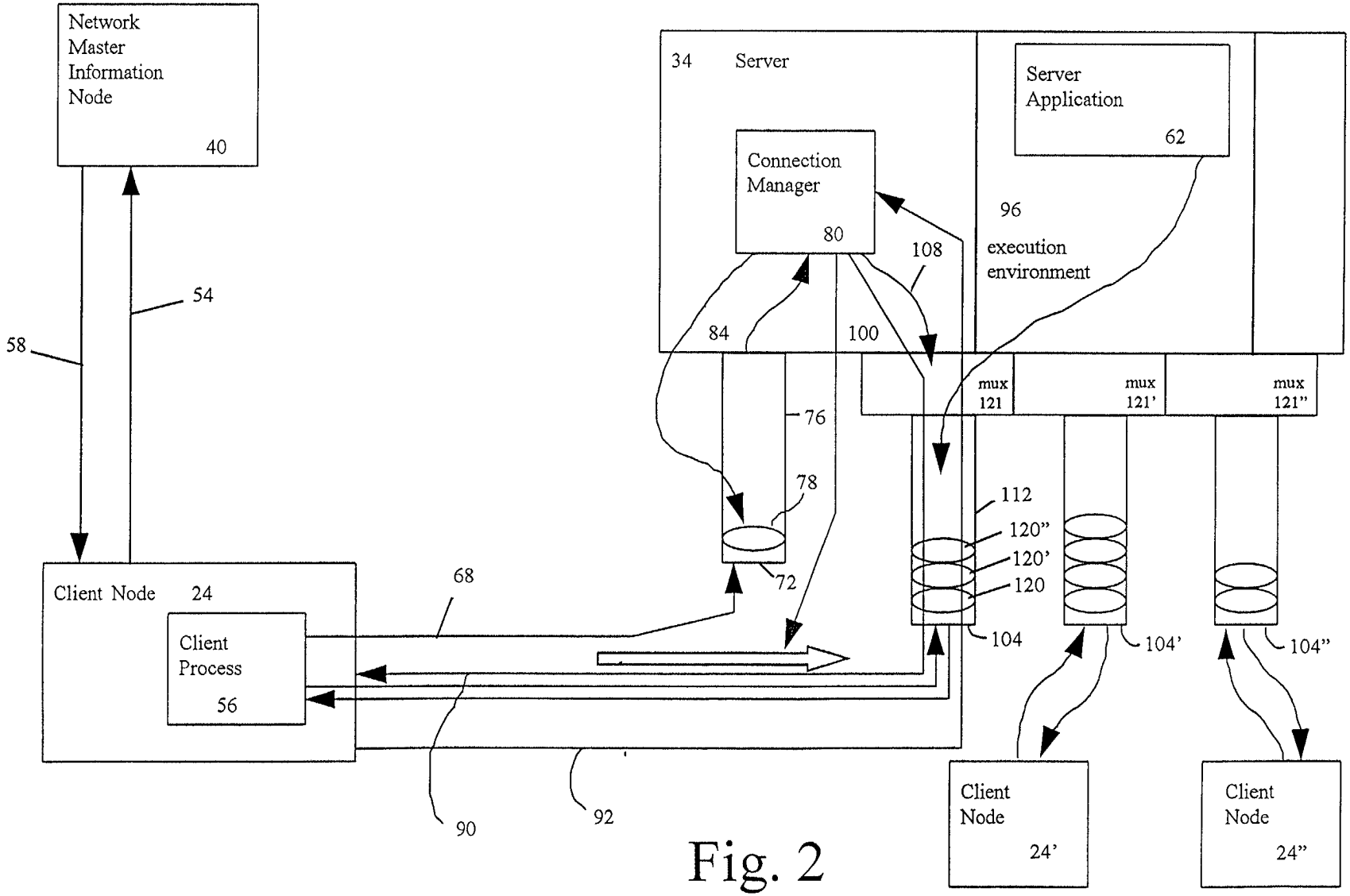

Fig. 2

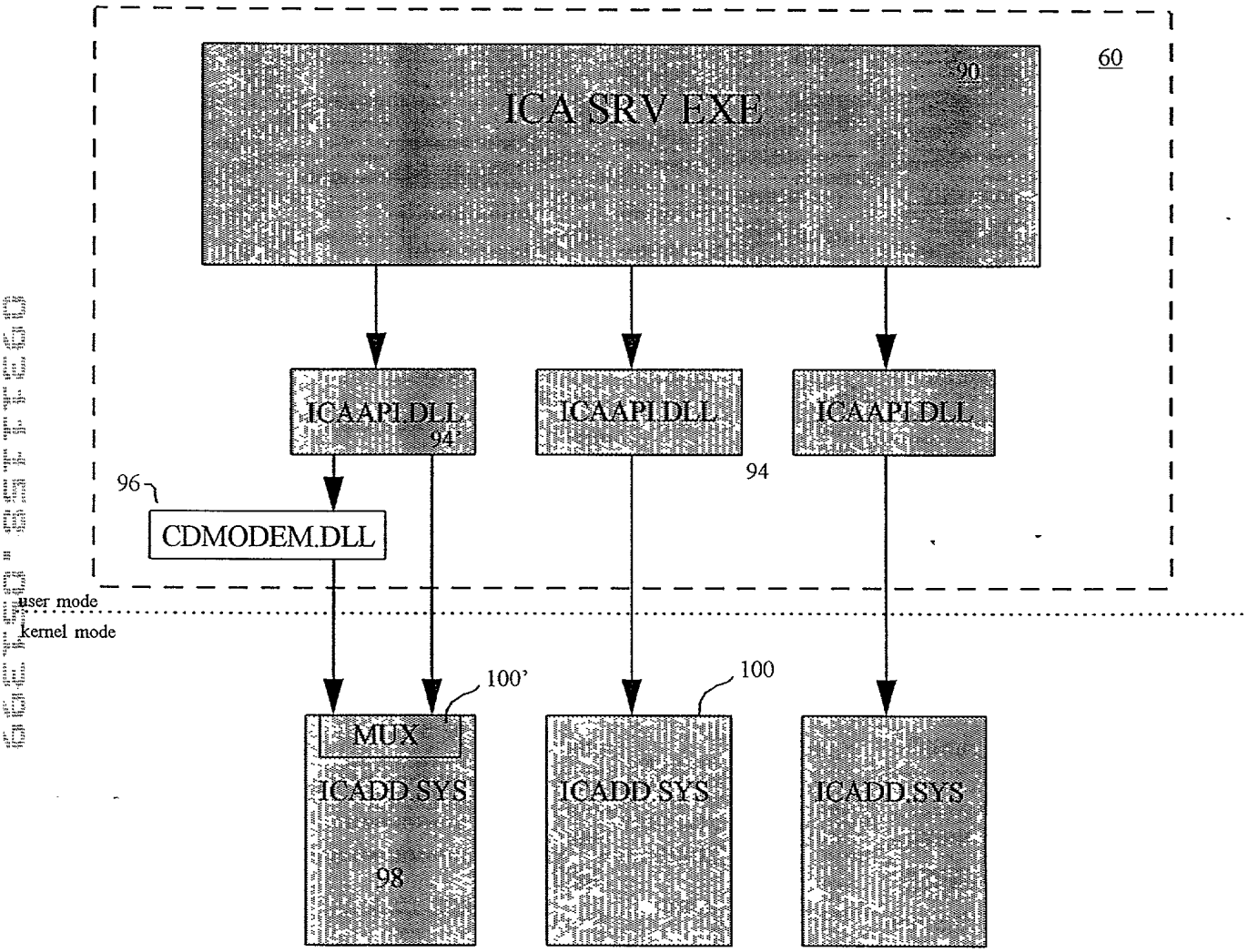
ICA SRV EXE
90
60
ICAAPI.DLL
94'
ICAAPI.DLL
ICAAPI.DLL
94
96
CDMODEM.DLL
user mode
kernel mode
100'
100
MUX
ICADD.SYS
98
ICADD.SYS
ICADD.SYS

Fig. 3

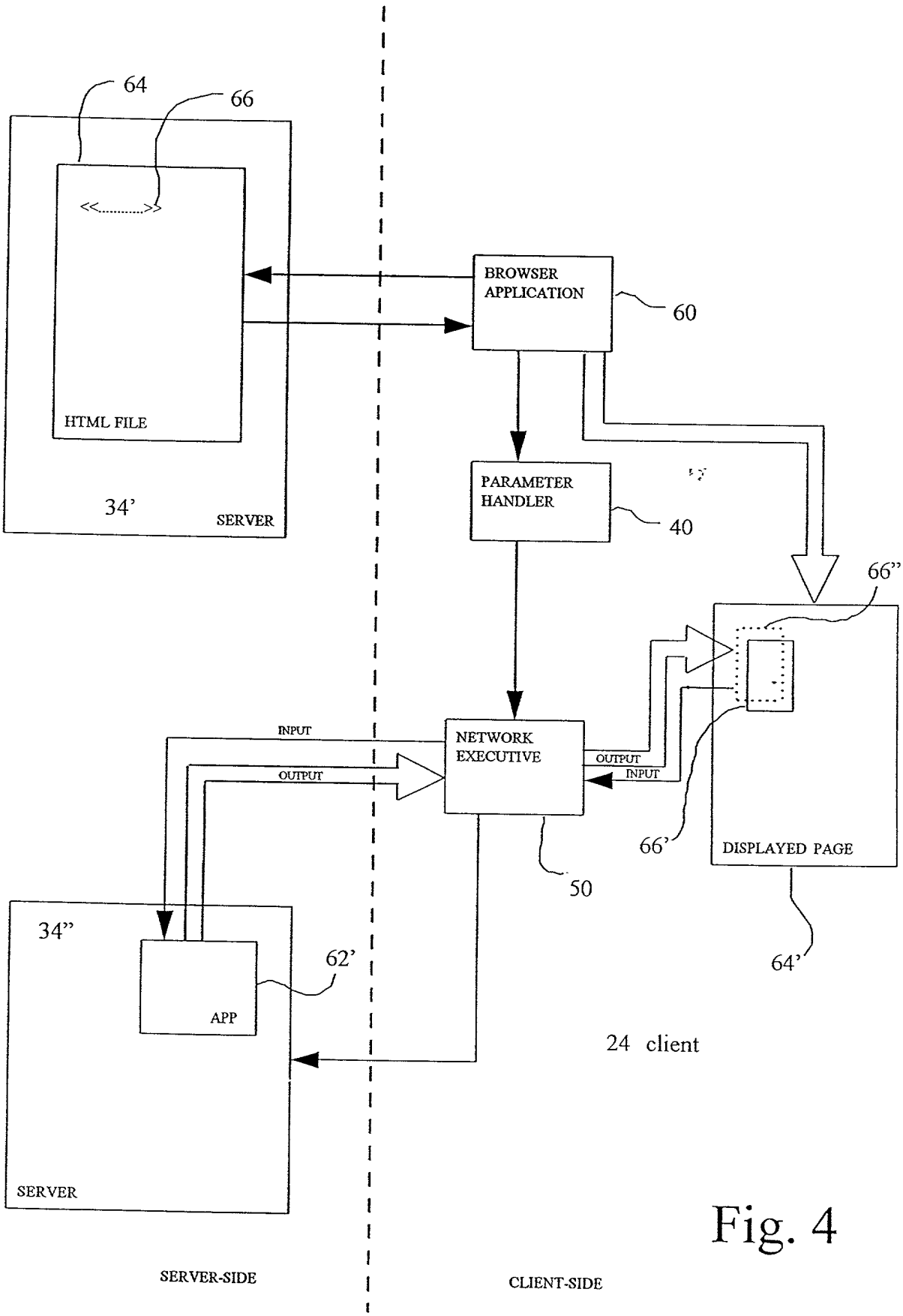
64
66
HTML FILE
34'
SERVER
BROWSER APPLICATION
60
PARAMETER HANDLER
40
66"
INPUT
OUTPUT
NETWORK EXECUTIVE
OUTPUT
INPUT
66'
DISPLAYED PAGE
50
64'
34"
62'
APP
SERVER
24 client
SERVER-SIDE
CLIENT-SIDE

Fig. 4

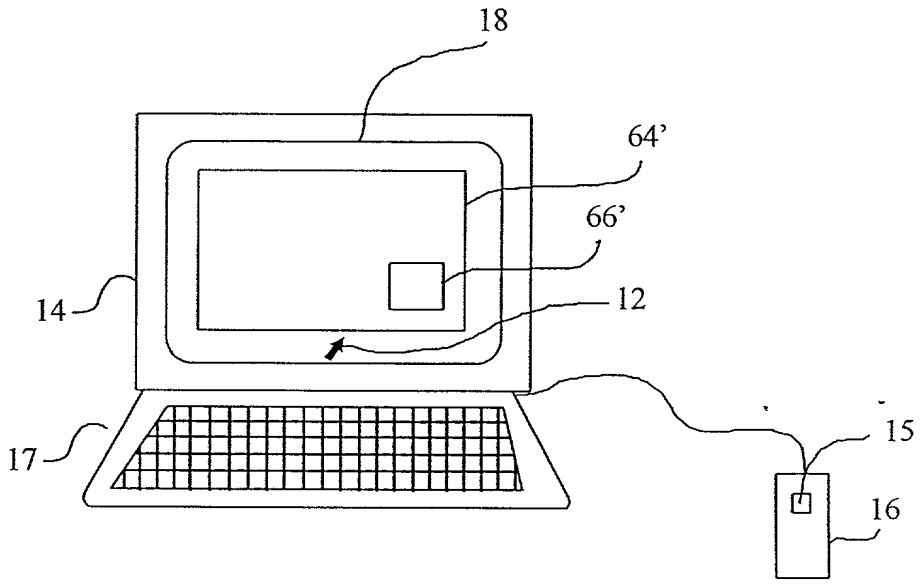

Fig. 5

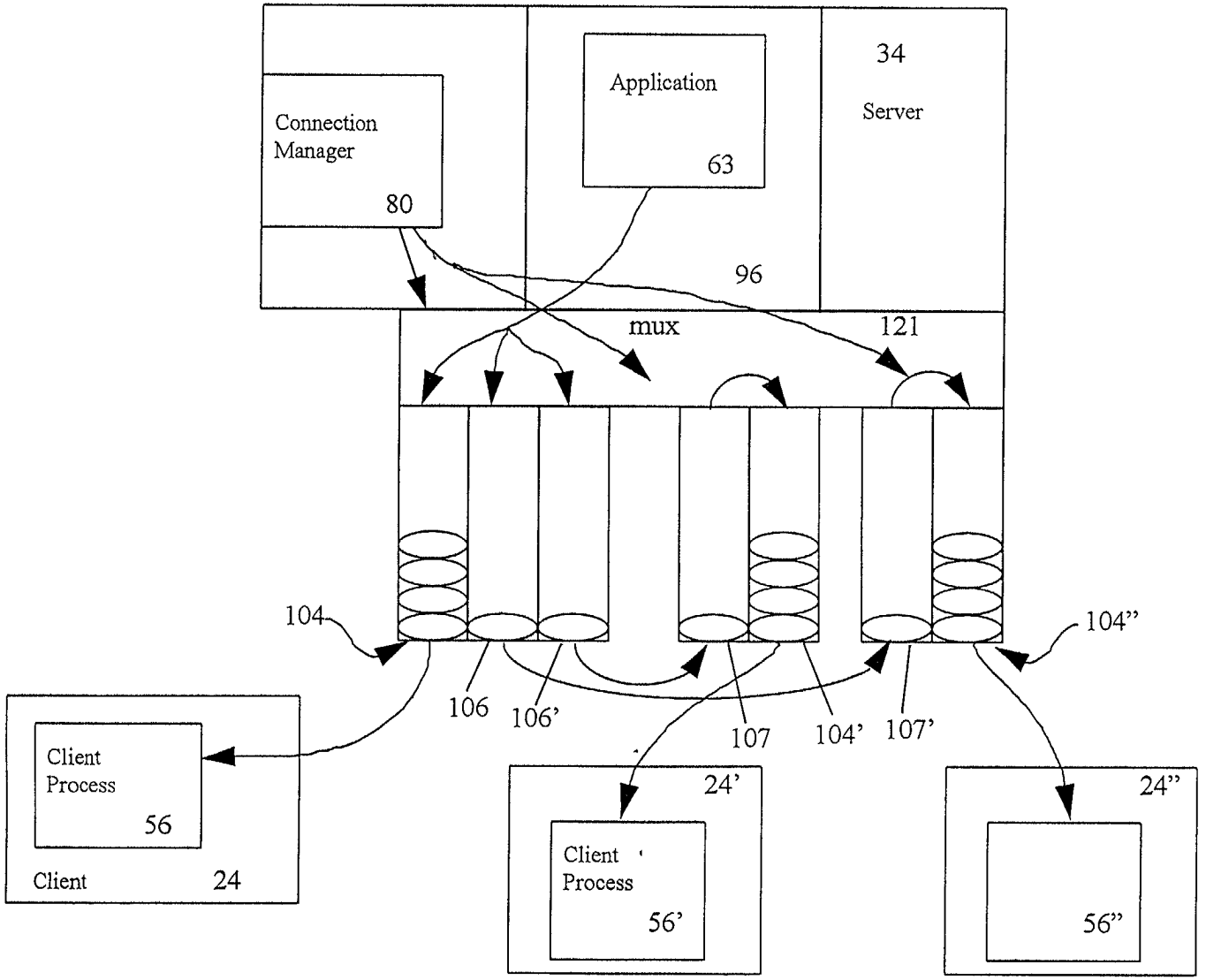

Fig. 6

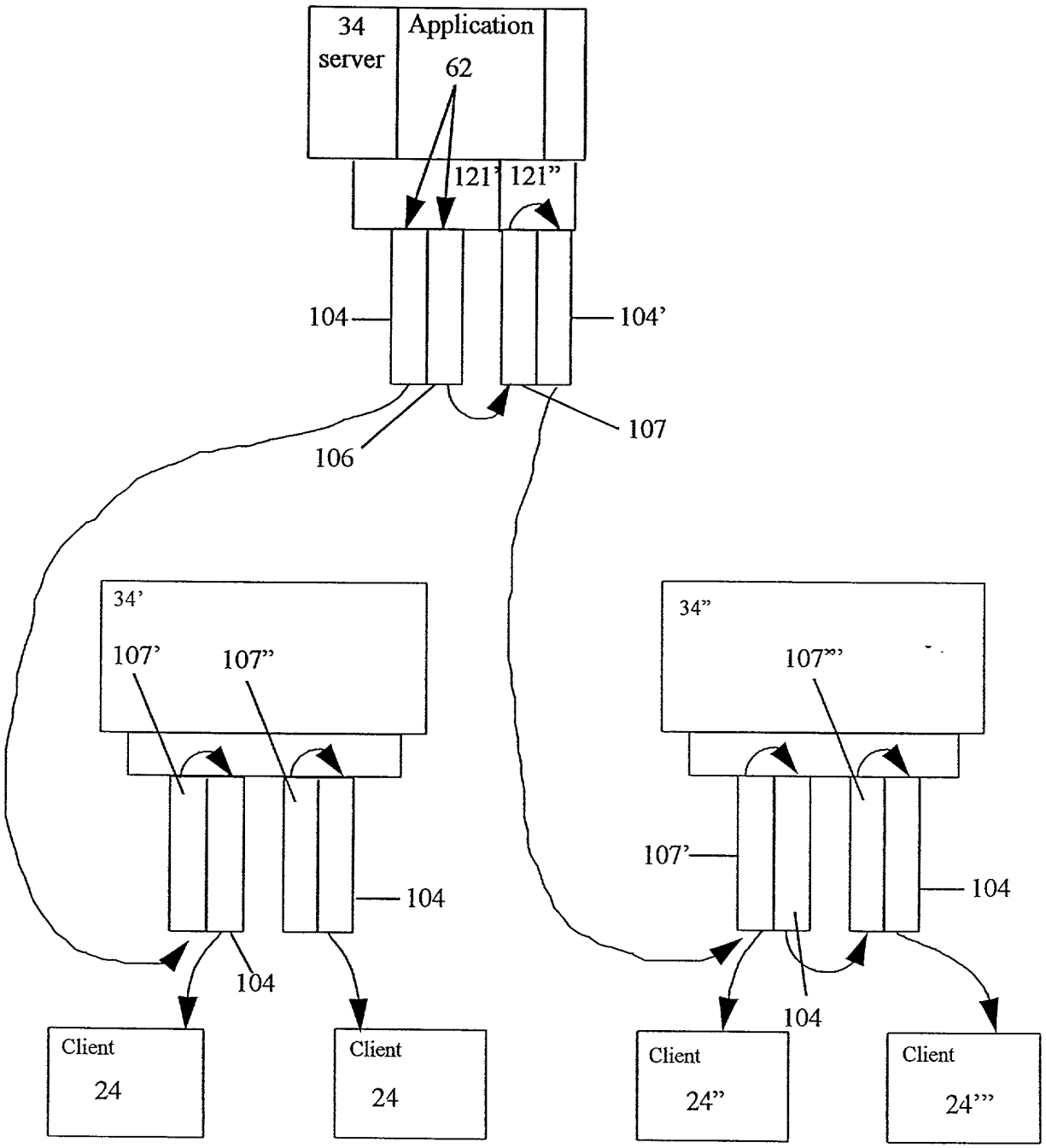
34
server
Application
62
121'
121"
104
104'
107
106
34'
107'
107"
104
104
34"
107"'
107'
104
104
Client
24
Client
24
Client
24"
Client
24"'

Fig. 7

EXPRESS MAIL LABEL NO. EM443447383US

PATENT
Atty. Docket No. CTX-016
(1545/27)

COMBINED DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

(Original, Design, National Stage of PCT, Supplemental, Divisional, Continuation or CIP)

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name, and I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

SYSTEM AND METHOD FOR TRANSMITTING DATA FROM A SERVER APPLICATION TO MORE THAN ONE CLIENT NODE

the specification of which (check one):

☒ is attached hereto.

☐ was filed on _____ as Application Serial No. 0 _____ / _____ or

☐ was described and claimed in PCT International Application No. _____ filed on _____ and as amended under PCT Article 19 on _____ (if any).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims as amended by any amendment referred to herein.

I acknowledge the continuing duty to disclose information which is material to the examination of this application in accordance with 37 C.F.R. §1.56.

PRIORITY CLAIM

☐ A. I hereby claim benefit under 35 U.S.C. 119(e) of United States Provisional Application No. ___, filed on _____.

☐ B. I hereby claim foreign priority benefits under 35 U.S.C. §119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and I have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed.

N

☐ no such applications have been filed.

☐ such applications have been filed as follows:

EARLIEST FOREIGN APPLICATION(S), IF ANY FILED WITHIN 12 MONTHS (6 MONTHS FOR DESIGN) PRIOR TO THIS U.S. APPLICATION

Country	Application Number	Date of Filing (mo., day, year)	Priority Claimed Under 35 USC 119
			☐YES NO ☐
			☐YES NO ☐
			☐YES NO ☐

☐ C. I hereby claim the benefit under 35 U.S.C. §120 of any United States application(s) or PCT international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in that/those prior application(s) in the manner provided by the first paragraph of 35 U.S.C. §112, I acknowledge the duty to disclose material information as defined in 37 C.F.R. §1.56 which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application.

PRIOR U.S. NON-PROVISIONAL APPLICATIONS OR PCT INTERNATIONAL APPLICATIONS DESIGNATING THE U.S. FOR BENEFIT UNDER 35 USC §120:

U.S. APPLICATIONS	U.S. FILING DATE	STATUS
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, aband.)
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, aband.)
(Application Serial No.)	(Filing Date)	(Status) (patented, pending, aband.)

POWER OF ATTORNEY

As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the United States Patent and Trademark Office connected therewith:

Steven M. Bauer	Reg. No. 31,481
Paula A. Campbell	Reg. No. 32,503
Joseph A. Capraro, Jr.	Reg. No. 36,471
John J. Cotter	Reg. No. 38,116
Gillian M. Fenton	Reg. No. 36,508
Duncan A. Greenhalgh	Reg. No. 38,678
Robin D. Kelley	Reg. No. 34,637
Douglas J. Kline	Reg. No. 35,574
John D. Lanza	Reg. No. 40,060
Robin R. Longo	Reg. No. 40,071
Thomas C. Meyers	Reg. No. 36,989
Edmund R. Pitcher	Reg. No. 27,829
Kurt Rauschenbach	Reg. No. 40,137
Scott J. Southworth	Reg. No. 39,382
Christopher W. Stamos	Reg. No. 35,370
Robert J. Tosti	Reg. No. 35,393
Thomas A. Turano	Reg. No. 35,722
Michael J. Twomey	Reg. No. 38,349
Christine C. Vito	Reg. No. 39,061

Direct correspondence to:

Patent Administrator
Testa, Hurwitz & Thibeault, LLP
High Street Tower
125 High Street
Boston, MA 02110

Direct telephone calls to:

John D. Lanza (617) 248-7604

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

SIGNATURE(S)

Bradley J. Pedersen	USA
Full name of inventor	Citizenship
Bradley J Peder	5/5/97
Inventor's signature	Date

7700 S. Woodridge Drive, Parkland, Florida 33067
Residence

Same
Post Office Address

412JDL1545/27.353160-1

EXPRESS MAIL LABEL NO. EM443447383US

PATENT

Attorney Docket No. CTX-016CN
(1545/27)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT:	Pedersen		
SERIAL NO.:	Not yet assigned	GROUP NO.:	Not yet assigned
FILING DATE:	Herewith	EXAMINER:	Not yet assigned
TITLE:	System and Method for Transmitting Data From a Server Application to More Than One Client Node		

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

ASSOCIATE POWER OF ATTORNEY

An associate power of attorney is hereby granted to:

NAME	REG. NO.
Michael A. Rodriguez	41,274

in connection with the above-identified patent application.

Please continue to direct all correspondence relating to the above application to:

Patent Administrator
Testa, Hurwitz & Thibeault, LLP
High Street Tower
125 High Street
Boston, MA 02110

Respectfully submitted,

Date: May 13, 1999
Reg. No. 40,060

John D. Lanza
Attorney for Applicant
Testa, Hurwitz, & Thibeault, LLP
High Street Tower
125 High Street
Boston, Massachusetts 02110

Tel. No.: (617) 248-7604
Fax No.: (617) 248-7100

412JDL1545/27.793013-1